2 Systems of ODE and higher-order ODEs

An m-dimensional system of differential equations generalizes (1.1) to

$$\vec{y}'(x) = \vec{f}(x, \vec{y}(x)) = \left(f_1(x, \vec{y}(x)), f_2(x, \vec{y}(x)), \cdots, f_m(x, \vec{y}(x))\right) \qquad \textbf{(2.1)}$$

with initial vector condition $\vec{y}(x_0) = \vec{y}_0$.

The arrows on top of the symbols y or f denotes an m-dimensional vector expression.

Example 2.1: Van der Pol second-order differential equation

The van der Pol equation emerges in the study of a closed loop electrical circuit consisting of an inductor, a capacitor, and a nonlinear resistor. It is a classical example of a nonconservative nonlinear system with a stable limit cycle.

$$z''(t) = \underbrace{\mu(1-z^2)}_{non-linear\ damping}\ z' - z \qquad \textbf{(2.2a)}$$

The r.h.s.-function of a differential equation $z^{(n)} = f(x, z, z', z'', \cdots, z^{(n-1)})$ with highest derivative of order n is $f(x, \underbrace{z, z_1, z_2, \cdots, z_{n-1}}_{=:\vec{z}})$. This is multi-variate in the z-coordinates.

Substituting $z_1 = z', z_2 = z'', \ldots, z_{n-1} = z^{(n-1)}$ transforms a higher order differential equation into a 1st – order system:

$$\boxed{\begin{aligned} z' &= z_1 \\ z_1' &= z_2 \\ z_2' &= z_3 \\ &\vdots \\ z_{n-1}' &= f(x, z, z_1, z_2, \cdots, z_{n-1}) \end{aligned}} \qquad \textbf{(2.3)}$$

with solution vector $\vec{y} = (z, z_1, z_2, \cdots, z_{n-1})$.

Example 2.1 cont: Applying this scheme to (2.2a) and then renaming z_1 with v yields:

$$\begin{pmatrix} z' = v \\ v' = \mu(1-z^2)v - z \end{pmatrix}. \qquad \textbf{(2.2b)}$$

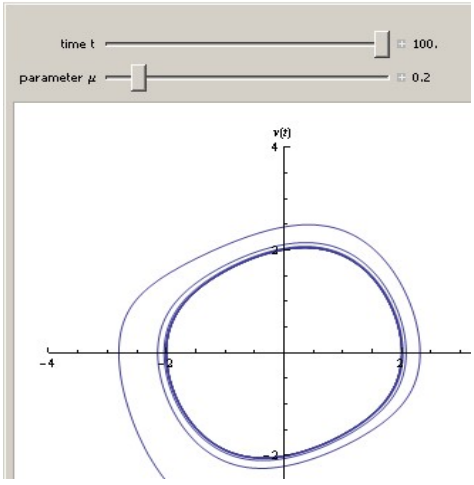

Figure 2.1: Phase plot of a solution $\vec{y}(t) = (z(t), \underbrace{z_1(t)}_{=z'(t)})$ to (2.2ab) for μ = 0.2 with indicated initial point. The solution (trajectory) has a limit cycle.

In the undamped case μ = 0 the solution curve is a circle.

For the special initial condition (0,0) the solution is constant 0 (circle with radius 0), but for any other initial condition the system has a limit cycle. Thus (0,0) is an unstable equilibrium point.

Example 2.2: 3D Lorenz equations

Lorenz derived the equations 1963 from a set of partial equations modelling convection rolls in the atmosphere. The equations constitute a 3D system of non-linear differential equations.

$$\begin{aligned} y_1'(t) &= f_1(t, y_1, y_2, y_3) = \sigma(y_2 - y_1) \\ y_2'(t) &= f_2(t, y_1, y_2, y_3) = y_1(\rho - y_3) - y_2 \\ y_3'(t) &= f_3(t, y_1, y_2, y_3) = y_1 y_2 - \beta y_3 \end{aligned} \qquad \textbf{(2.4)}$$

The parameters σ, β, ρ are positive > 0, but usually $\sigma = 10, \beta = 8/3$ and ρ is varied. For small values of ρ the system is stable and evolves two one of two fixed point attractors. For ρ = 28, e.g., the system exhibits (complex) chaotic behaviour.

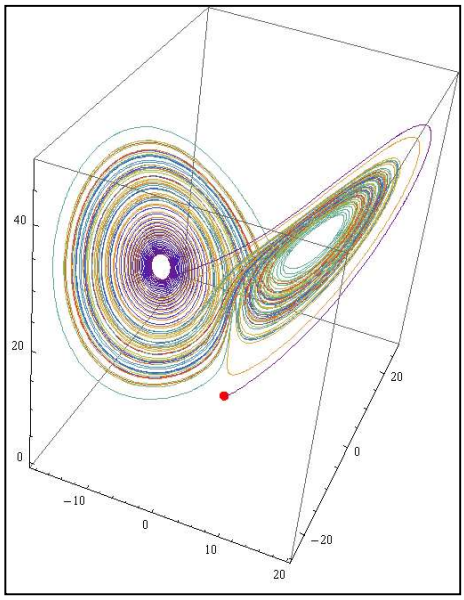

Figure 2.2: The 3D-plot on the left shows the evolution of the solution vector $\vec{y}(t)$ with (red) inital point $\vec{y}(0)$ = (0,0,1).

Here ρ = 28 and $0 \le t \le 200$. The system was solved numerically with the embedded Runge-Kutta method Bogacki-Shampine 5(4) with accuracy and precision goals equal to 16.

Example 2.3: Planar three-body-problem

The trajectories of n = 3 mass points with masses $m_i > 0 \quad (i = 1,2,...,n = 3)$ and planar coordinates $\vec{q}_i(t) = \left(x_i(t), y_i(t)\right) \quad (i = 1,2,...,n = 3)$ are described by a system of Newton equations (second law of motion with gravitational constant γ):

$$m_i \vec{q}_i''(t) = \gamma \sum_{k=1, k \neq i}^{n} \frac{m_i m_k (\vec{q}_k - \vec{q}_i)}{\left|\vec{q}_k - \vec{q}_i\right|^3} \quad (i = 1,2,...,n = 3) \qquad \textbf{(2.5a)}$$

The r.h.s. expresses the vector sum of all gravitational forces applied to the *i*-th body due to all other bodies.

$$
\begin{aligned}
\frac{1}{\gamma} m_1 x_1''(t) &= -\frac{m_1 m_2 (x_1 - x_2)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}^3} - \frac{m_1 m_3 (x_1 - x_3)}{\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}^3} \\
\frac{1}{\gamma} m_1 y_1''(t) &= -\frac{m_1 m_2 (y_1 - y_2)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}^3} - \frac{m_1 m_3 (y_1 - y_3)}{\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}^3} \\
\frac{1}{\gamma} m_2 x_2''(t) &= -\frac{m_1 m_2 (x_2 - x_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}^3} - \frac{m_2 m_3 (x_2 - x_3)}{\sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}^3} \\
\frac{1}{\gamma} m_2 y_2''(t) &= -\frac{m_1 m_2 (y_2 - y_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}^3} - \frac{m_2 m_3 (y_2 - y_3)}{\sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}^3} \\
\frac{1}{\gamma} m_3 x_3''(t) &= -\frac{m_1 m_3 (x_3 - x_1)}{\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}^3} - \frac{m_2 m_3 (x_3 - x_2)}{\sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}^3} \\
\frac{1}{\gamma} m_3 y_3''(t) &= -\frac{m_1 m_3 (y_3 - y_1)}{\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}^3} - \frac{m_2 m_3 (y_3 - y_2)}{\sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}^3}
\end{aligned}
\qquad \textbf{(2.5b)}
$$

This is a system of 6 coupled second-order equations.

Introducing the (velocities) variables $\left(v_{x_i}(t) := x_i'(t), v_{y_i}(t) = y_i'(t)\right) \quad (i = 1,2,3)$ a first-order system with 12 equations results when applying the scheme (2.3).

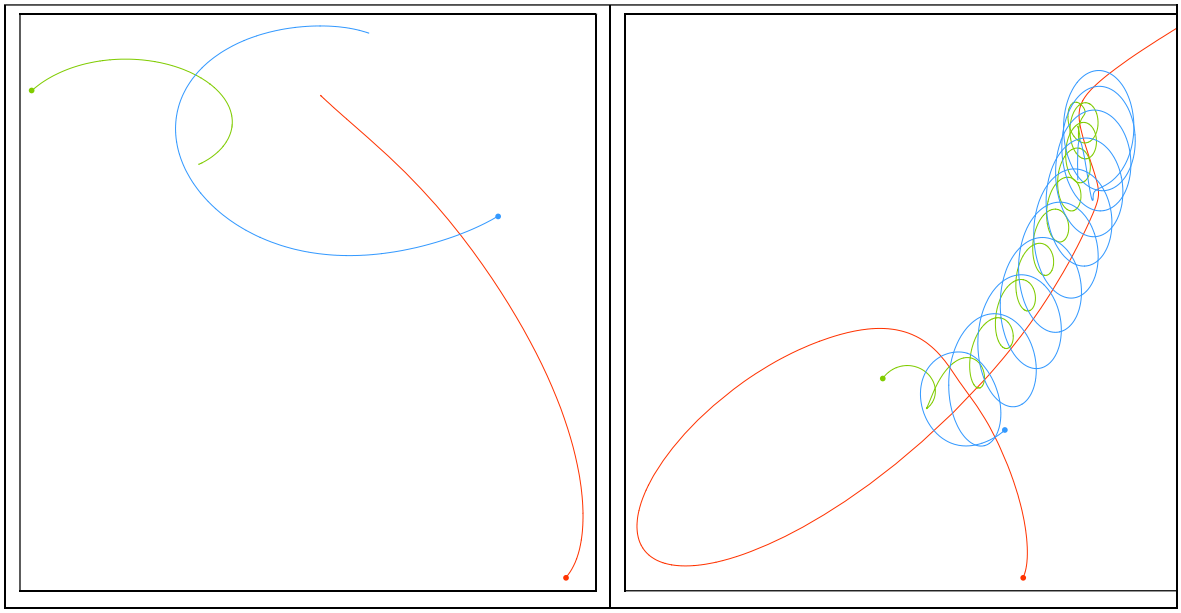

Figure 2.3ab: Planar trajectories of 3 bodies. The plots were generated with the embedded Runge-Kutta method Bogacki-Shampine 5(4) with accuracy and precision goal set to 16. The first picture shows an evolution until t = 0.67 sec and the second (scaled) one (the same evolution) until t = 10 sec.

2.1 Notes on generalizations from scalar functions to vector-valued functions

The theory on explicit methods developped in the previous chapter can be generalized to systems of differential equations. Concepts, proofs and formulaes can be carried over literally or with slight adaptations in notation mostly by substituting y with $\vec{y} = \vec{y}(x)$, i.e. a vector-valued function depending on the scalar variable x.

The multi-variate chain rule played a fundamental role in Property 1.1. Therefore it will be demonstrated here in vector notation:

$$\vec{y}''(x) = \frac{d\vec{f}}{dx} = J_{\vec{f}}(x,\vec{y}) \cdot (1, \vec{y}'(x))^{tr} = \left(\frac{\partial \vec{f}}{\partial x}, \underbrace{\frac{\partial \vec{f}}{\partial y_1}, \frac{\partial \vec{f}}{\partial y_2}, \cdots, \frac{\partial \vec{f}}{\partial y_m}}_{=:\frac{\partial \vec{f}}{\partial \vec{y}} = J_{\vec{f}}(\vec{y})} \right) \cdot \left(1, \underbrace{y_1'(x), y_2'(x), \cdots, y_m'(x)}_{=:\vec{y}'(x)} \right)^{tr} =$$

$$\frac{\partial \vec{f}}{\partial x} + \frac{\partial \vec{f}}{\partial \vec{y}} \cdot \vec{y}'(x) := \vec{f}_x + \vec{f}_{\vec{y}} \cdot \vec{f} := \vec{F}_1$$

The expressions $\frac{\partial \vec{f}}{\partial \vec{y}} = J_{\vec{f}}(\vec{y}) = \vec{f}_{\vec{y}}$ all denote the Jacobian matrix of $\vec{f}$ with respect to the $\vec{y}$-variables $(y_1, y_2, \cdots, y_m)$. So it becomes clear that even the abbreviations introduced in (1.8ab) carry over. The dot here denotes the product of matrices with matrices or vectors (the inner product being a special case). Keeping in mind that the product rule generalizes literally to products of matrices with vectors higher derivatives $\vec{y}'''(x), \vec{y}^{(4)}(x), \ldots$ are computed by iterative application of the multi-variate chain rule combined with the product rule:

$$\vec{y}'''(x) = \frac{d}{dx}\left(\underbrace{\vec{f}_x + \vec{f}_{\vec{y}} \cdot \vec{f}}_{=:\vec{F}_1} \right) = \frac{d}{dx}\vec{f}_x + \frac{d}{dx}\left(\vec{f}_{\vec{y}} \cdot \vec{f}\right) = \vec{f}_{xx} + \vec{f}_{x\vec{y}} \cdot \vec{f} + \left(\frac{d}{dx}\vec{f}_{\vec{y}}\right) \cdot \vec{f} + \vec{f}_{\vec{y}} \cdot \left(\frac{d}{dx}\vec{f}\right) =$$

$$\vec{f}_{xx} + \vec{f}_{x\vec{y}} \cdot \vec{f} + \left(\vec{f}_{\vec{y}x} + \vec{f}_{\vec{y}\vec{y}} \cdot \vec{f}\right) \cdot \vec{f} + \vec{f}_{\vec{y}} \cdot \left(\vec{f}_x + \vec{f}_{\vec{y}} \cdot \vec{f}\right) =$$

$$\vec{f}_{xx} + 2\vec{f}_{x\vec{y}} \cdot \vec{f} + \vec{f}_{\vec{y}} \cdot \vec{f}_x + \vec{f}_{\vec{y}}{}^2 \cdot \vec{f} + \left(\vec{f}_{\vec{y}\vec{y}} \cdot \vec{f}\right) \cdot \vec{f} =$$

$$\underbrace{\vec{f}_{xx} + 2\vec{f}_{x\vec{y}} \cdot \vec{f} + \left(\vec{f}_{\vec{y}\vec{y}} \cdot \vec{f}\right) \cdot \vec{f}}_{=:\vec{F}_2} + \vec{f}_{\vec{y}} \cdot \vec{f}_x + \vec{f}_{\vec{y}}{}^2 \cdot \vec{f} = \vec{F}_2 + \vec{f}_{\vec{y}} \cdot \vec{F}_1$$

provided that all second partial derivatives exist and are continuous (from the continuity it follows the symmetry relation $\vec{f}_{x\vec{y}} = \vec{f}_{\vec{y}x}$ by a theorem called Schwarz-theorem). The expression $\vec{f}_{\vec{y}\vec{y}}$ is a kind of „matrix gradient“ $\left(\frac{\partial}{\partial y_1}\vec{f}_{\vec{y}}, \frac{\partial}{\partial y_m}\vec{f}_{\vec{y}}, \cdots, \frac{\partial}{\partial y_m}\vec{f}_{\vec{y}} \right)$ whose components are partial derivatives of Jacobian matrices and thus matrices again. Then $\left(\vec{f}_{\vec{y}\vec{y}} \cdot \vec{f}\right)$ is a linear combination of matrices and $\left(\vec{f}_{\vec{y}\vec{y}} \cdot \vec{f}\right) \cdot \vec{f}$ is an m-dimensional vector.

A second generalization concerns Taylor-expansions of $\vec{f}$ with respect to multiple y-coordinates and multiple components. This is important because such expansions occur in the derivation of Runge-Kutta methods (as in (1.8b) where the k-values are expanded according to Taylor's formula). Instead of scalar k-values k-vectors enter the stage. The idea becomes clear when generalizing (1.8):

Now we are to find coefficients a, b and m, n such that the (so-called 2-stage Runge-Kutta) formulas

$$\left.\begin{aligned}\vec{k}_1 &= h\,\vec{f}(x,\vec{y})\\ \vec{k}_2 &= h\,\vec{f}(x+mh,\vec{y}+m\vec{k}_1)\end{aligned}\right\}\ 2\ stages$$

$$\vec{y}(x+h) \approx \vec{y}(x) + a\vec{k}_1 + b\vec{k}_2 \qquad \textbf{(2.6)}$$

duplicate the Taylor series of the solution $\vec{y}$ through the (second-order) term in h^2. The last formula of (2.6) is then near the Taylor polynomial of order 2.

Introducing the usual abbreviations for $\vec{f}$ and its partial derivatives we get from above:

$$\vec{y}(x+h) \approx \vec{y}(x) + \vec{f}\,h + \frac{1}{2!}\underbrace{\left(\vec{f}_x + \vec{f}_{\vec{y}}\cdot\vec{f}\right)}_{\vec{F}_1}h^2 \qquad \textbf{(2.6a)}$$

Multi-variate Taylor expansions for the $\vec{k}$ -vectors produce (!):

$$\begin{aligned}\vec{k}_1 &= h\vec{f}\\ \vec{k}_2 &= h\left(\vec{f} + \vec{f}_x mh + \vec{f}_{\vec{y}}\cdot m\vec{k}_1 + \frac{1}{2!}\vec{f}_{xx}m^2h^2 + \vec{f}_{x\vec{y}}\cdot(mh)(m\vec{k}_1) + \frac{1}{2!}\left(\vec{f}_{\vec{y}\vec{y}}m^2\vec{k}_1\right)\cdot\vec{k}_1 + \cdots\right)\\ &= h\left(\vec{f} + \vec{f}_x mh + \vec{f}_{\vec{y}}\cdot m\vec{k}_1 + \frac{1}{2!}\vec{f}_{xx}m^2h^2 + \vec{f}_{x\vec{y}}\cdot(mh)(mh\vec{f}) + \frac{1}{2!}\left(\vec{f}_{\vec{y}\vec{y}}m^2h\vec{f}\right)\cdot h\vec{f} + \cdots\right) \qquad \textbf{(2.6b)}\\ &= h\left(\vec{f} + m\underbrace{\left(\vec{f}_x + \vec{f}_{\vec{y}}\cdot\vec{f}\right)}_{\vec{F}_1}h + \frac{1}{2!}m^2\left(\underbrace{\vec{f}_{xx} + 2\vec{f}_{x\vec{y}}\cdot\vec{f} + \left(\vec{f}_{\vec{y}\vec{y}}\cdot\vec{f}\right)\cdot\vec{f}}_{=:\vec{F}_2}\right)h^2 + \cdots\right)\end{aligned}$$

Combining the expressions in (2.6b) as suggested in (2.6a) yields

$$\vec{y}(x+h) \approx \vec{y}(x) + (a+b)h\vec{f} + (bm)\vec{F}_1h^2\,.$$

Comparing coefficients with (1.8a rev.) then must give the same redundant system of equations as in (1.8c): $a+b=1$, $bm=\frac{1}{2}$.

From this it becomes clear that the framework for Runge-Kutta methods (including Butcher tableaus) can be carried over to systems of differential equations. The next example is a demonstration.

Example 2.4: Heun method applied to van der Pol's equation (Ex. 2.1 cont.)

The van der Pol equation $\begin{pmatrix} z' = v \\ v' = \mu(1-z^2)v - z\end{pmatrix}$ (cf. 2.2b) has as r.h.s.-function

$$\vec{f}(t,z,v) = \begin{pmatrix} f_1(t,z,v) = & v \\ f_2(t,z,v) = & \mu(1-z^2)v - z\end{pmatrix}.$$

So the Heun-method (cf. 1.9a) as a 2-stage Runge-Kutta method with $\vec{y} = (z,v)$ reads as:

$$\left.\begin{aligned}
\vec{k}_1 &= h\,\vec{f}(t,z,v) = \begin{pmatrix} h\,f_1(t,z,v) = & h\,v \\ h\,f_2(t,z,v) = & h\left(\mu(1-z^2)v - z\right) \end{pmatrix} \\
\vec{k}_2 &= h\,\vec{f}(t+h,\vec{y}+\vec{k}_1) = \begin{pmatrix} h\,f_1\left(t+h, z+h\,v, v+h\left(\mu(1-z^2)v-z\right)\right) \\ h\,f_2\left(t+h, z+h\,v, v+h\left(\mu(1-z^2)v-z\right)\right) \end{pmatrix} \\
&= \begin{pmatrix} h\left(v+h\left(\mu(1-z^2)v-z\right)\right) \\ h\left(\mu(1-(z+h\,v)^2)\left(v+h\left(\mu(1-z^2)v-z\right)\right)-(z+h\,v)\right) \end{pmatrix}
\end{aligned}\right\} \; 2\ stages$$

$$\vec{y}(t+h) \approx \vec{y}(t) + \frac{1}{2}\vec{k}_1 + \frac{1}{2}\vec{k}_2$$

A short (modern) vector notation with explicit Butcher tableau $\begin{pmatrix} 0 & 0 & 0 \\ c_2 & a_{21} & 0 \\ & b_1 & b_2 \end{pmatrix}$ is:

$$\left.\begin{aligned}
\vec{k}_1 &= \vec{f}(t,\vec{y}) \\
\vec{k}_2 &= \vec{f}(t+c_1 h, \vec{y}+a_{21}\vec{k}_1)
\end{aligned}\right\} \; 2\ stages$$

$$\vec{y}(t+h) \approx \vec{y}(t) + b_1 h\vec{k}_1 + b_2 h\vec{k}_2$$

The application of adaptive step-size formulas like (1.13d) in the case of vector-valued functions is described in detail in the footnote to (1.13d).

t_k	Heun-Euler 2(1)	Reference
0.	{1., -1.}	{1., -1.}
0.11	{0.88395, -1.10649}	{0.884085, -1.10632}
0.208365	{0.7706, -1.19517}	{0.770853, -1.1949}
0.298483	{0.659371, -1.27057}	{0.659728, -1.27025}
0.382377	{0.549951, -1.33523}	{0.550398, -1.33493}
0.461389	{0.442153, -1.39071}	{0.442707, -1.39038}
0.536474	{0.335855, -1.43798}	{0.336514, -1.43762}
0.608352	{0.23097, -1.47764}	{0.231647, -1.47734}
0.67759	{0.127436, -1.51008}	{0.128093, -1.50985}
0.744656	{0.0252065, -1.5355}	{0.0258871, -1.53532}
0.810278	{-0.0762713, -1.55408}	{-0.0754934, -1.55394}
0.875774	{-0.178556, -1.56582}	{-0.177612, -1.56573}
0.941269	{-0.281378, -1.57019}	{-0.280245, -1.57019}
1.00685	{-0.384365, -1.56662}	{-0.383101, -1.56673}
1.0727	{-0.487282, -1.55453}	{-0.48592, -1.55418}
1.1391	{-0.589947, -1.53327}	{-0.588456, -1.53198}
1.20635	{-0.692184, -1.50212}	{-0.690539, -1.49989}
1.27484	{-0.793797, -1.46025}	{-0.791999, -1.4575}
1.34499	{-0.894546, -1.40671}	{-0.89263, -1.40413}
1.41733	{-0.994119, -1.34041}	{-0.992159, -1.33878}

Table 2.1: The first twenty steps from embedded pair Heun-Euler 2(1) with μ = 0.2. The accuracy goal is set to 1 and the precision goal to 2.

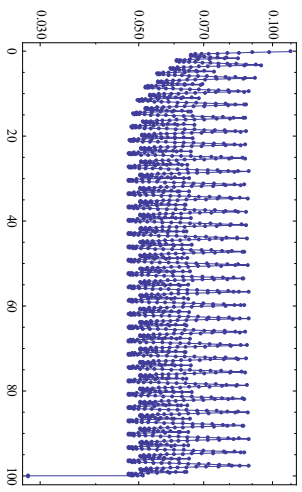

Figure 2.4: Plot of the Step-sizes

The norm function for step-size adaptation in Table 2.1 was set to the infinity norm (maximum norm). So the classical step-size formula (1.13de) for the example in Table 2.1 becomes

$$h_{n+1} = h_n \max\left\{ \frac{\left|\vec{e}_n^{(1)}\right|}{\varepsilon_a + \varepsilon_r \left|\vec{y}_n^{(1)}\right|}, \frac{\left|\vec{e}_n^{(2)}\right|}{\varepsilon_a + \varepsilon_r \left|\vec{y}_n^{(2)}\right|} \right\}^{-\frac{1}{\tilde{p}}} .$$

Here n denotes the step number and the superscripts in round brackets indicate the vector components.

Furthermore, $\vec{e}_n = (\vec{e}_n^{(1)}, \vec{e}_n^{(2)})$ is the vector of error estimates, $\vec{y}_n = (\vec{y}_n^{(1)} = z_n, \vec{y}_n^{(2)} = v_n)$ is the solution vector, and here $\tilde{p} = 2$ is the order of the (primary) Heun-method. Finally, the accuracy and precision goals determine $\varepsilon_a = 10^{-1}$, $\varepsilon_r = 10^{-2}$.

Example 2.5: Higher-order methods applied to van der Pol's equation (Ex. 2.1 cont.)

Table 2.2: Comparisons of the methods Bogacki-Shampine 5(4) and Dormand-Prince 5(4) for accuracy and precision goals 16. The second number in the step-column is the number of rejected steps. The costs-column indicates the numbers of function evaluations and the error-column indicates the miminum of final (global) relative or absolute error.

Method	Steps	Costs	Error
DP54	{37376, 26235}	381668	1.51545×10^{-14}
BS54	{23988, 26049}	350261	2.66454×10^{-15}

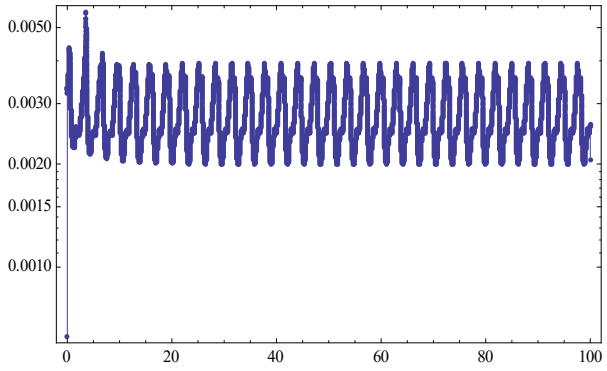

Figure 2.5a: Log-Plot of step-sizes for DP5(4)

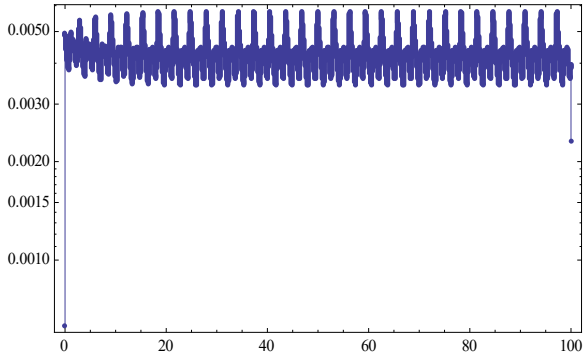

Figure 2.5b: Log-Plot of step-sizes for BS5(4)

Example 2.6: Higher-order methods applied to the Lorenz equations (Ex. 2.2 cont.)

The Lorenz equations show chaotic behaviour for many parameter settings. Usually $\sigma = 10, \beta = 8/3$ and ρ is varied. For small values of ρ the system is stable and evolves to one of two fixed point attractors. For ρ > 24.28, the fixed points become repulsors and the trajectory is repelled by them in a complex way, evolving without ever crossing itself. The chaotic behaviour means that (small) errors during a numeric scheme may evolve to divergent relative errors in the long run. This becomes visible in the next comparison with large errors relative to the reference solution (which itself is not reliable).

Method	Steps	Costs	Error
BS54	{251 668, 258 788}	3 573 194	0.757335

Method	Steps	Costs	Error
DP54	{386 656, 325 115}	4 270 628	1.1274

Tables 2.3ab: Comparisons of the methods Bogacki-Shampine 5(4) and Dormand-Prince 5(4) for accuracy and precision goals 16. Here the time interval was restricted to $0 \le t \le 100$. The second number in the step-column is the number of rejected steps. The costs-column indicates the number of function evaluations and the error-column indicates the minimum of final (global) relative or absolute error. The reference solution was computed with stiffness switching and a working precision of 32 for internal computations. Nevertheless, it is not necessarily reliable.

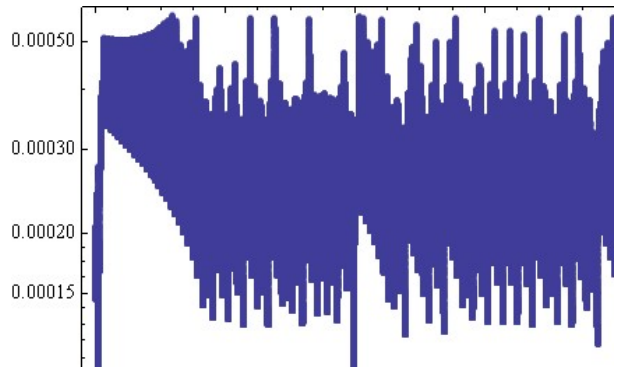

Figure 2.5a: Log-Plot of step-sizes for DP5(4)

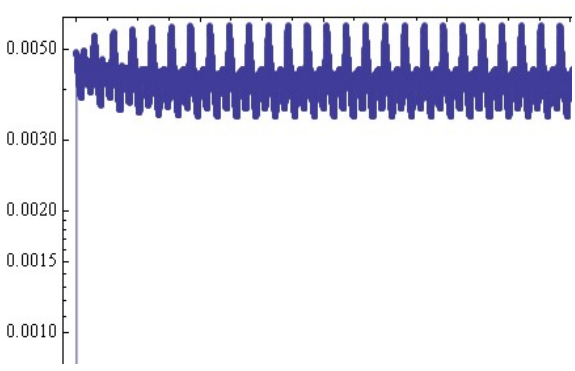

Figure 2.5b: Log-Plot of step-sizes for BS5(4)